python / **cpython** Public

Code Issues 5k+ Pull requests 2.2k Actions Projects Security and q

# Commit 8392b2f

**miss-islington** and **serhiy-storchaka** authored on Oct 8, 2025 · ✕ 57 / 69 · Verified

```
[3.12] gh-139700: Check consistency of the zip64 end of central directory record
(GH-139702) (GH-139708) (GH-139712)

(cherry picked from commit 333d4a6)

Support records with "zip64 extensible data" if there are no bytes
prepended to the ZIP file.
(cherry picked from commit 162997b)

Co-authored-by: Serhiy Storchaka <storchaka@gmail.com>
```

3.12 (#139712) · v3.12.13 v3.12.12

1 parent dea7e3d commit 8392b2f

## 3 files changed +113 -23 lines changed

Top

Filter files...

- Lib
  - test/test_zipfile
    - test_core.py
  - zipfile
    - __init__.py
- Misc/NEWS.d/next/Security
  - 2025-10-07-19-31-34.gh-issue-139700.vNHU1O.rst

## 3 files changed +113 -23 lines changed

Search within code

Lib/test/test_zipfile/test_core.py

```
@@ -885,6 +885,8 @@ def make_zip64_file(
```

```
885  885              self, file_size_64_set=False, file_size_extra=False,
886  886              compress_size_64_set=False, compress_size_extra=False,
887  887              header_offset_64_set=False, header_offset_extra=False,
     888  +           extensible_data=b'',
     889  +           end_of_central_dir_size=None, offset_to_end_of_central_dir=None,
888  890          ):
889  891              """Generate bytes sequence for a zip with (incomplete) zip64 data.
890  892
     @@ -938,6 +940,12 @@ def make_zip64_file(
938  940
939  941              central_dir_size = struct.pack('<Q', 58 + 8 *
     len(central_zip64_fields))
940  942              offset_to_central_dir = struct.pack('<Q', 50 + 8 *
     len(local_zip64_fields))
     943  +           if end_of_central_dir_size is None:
     944  +               end_of_central_dir_size = 44 + len(extensible_data)
     945  +           if offset_to_end_of_central_dir is None:
     946  +               offset_to_end_of_central_dir = (108
     947  +                                               + 8 * len(local_zip64_fields)
     948  +                                               + 8 * len(central_zip64_fields))
941  949
942  950              local_extra_length = struct.pack("<H", 4 + 8 *
     len(local_zip64_fields))
943  951              central_extra_length = struct.pack("<H", 4 + 8 *
     len(central_zip64_fields))
     @@ -966,14 +974,17 @@ def make_zip64_file(
966  974                  + filename
967  975                  + central_extra
968  976                  # Zip64 end of central directory
969       -              + b"PK\x06\x06,\x00\x00\x00\x00\x00\x00\x00-\x00-"
970       -              + b"\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00"
     977  +              + b"PK\x06\x06"
     978  +              + struct.pack('<Q', end_of_central_dir_size)
     979  +              + b"-\x00-
     \x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00"
971  980                  + b"\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
972  981                  + central_dir_size
973  982                  + offset_to_central_dir
```

```
                983 +             + extensible_data
974             984               # Zip64 end of central directory
975             -               + b"PK\x06\x07\x00\x00\x00\x00l\x00\x00\x00\x00\x00\x00\x00\x01"
976             -               + b"\x00\x00\x00"
                985 +             + b"PK\x06\x07\x00\x00\x00\x00"
                986 +             + struct.pack('<Q', offset_to_end_of_central_dir)
                987 +             + b"\x01\x00\x00\x00"
977             988               # end of central directory
978             989               + b"PK\x05\x06\x00\x00\x00\x00\x01\x00\x01\x00:\x00\x00\x002\x00"
979             990               + b"\x00\x00\x00\x00"
                  @@ -1004,6 +1015,7 @@ def test_bad_zip64_extra(self):
1004            1015              with self.assertRaises(zipfile.BadZipFile) as e:
1005            1016                  zipfile.ZipFile(io.BytesIO(missing_file_size_extra))
1006            1017              self.assertIn('file size', str(e.exception).lower())
                1018 +
                  self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_file_size_extra)))
1007            1019
1008            1020              # zip64 file size present, zip64 compress size present, one field in
1009            1021              # extra, expecting two, equals missing compress size.
                  @@ -1015,6 +1027,7 @@ def test_bad_zip64_extra(self):
1015            1027              with self.assertRaises(zipfile.BadZipFile) as e:
1016            1028                  zipfile.ZipFile(io.BytesIO(missing_compress_size_extra))
1017            1029              self.assertIn('compress size', str(e.exception).lower())
                1030 +
                  self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_compress_size_extra)))
1018            1031
1019            1032              # zip64 compress size present, no fields in extra, expecting one,
1020            1033              # equals missing compress size.
                  @@ -1024,6 +1037,7 @@ def test_bad_zip64_extra(self):
1024            1037              with self.assertRaises(zipfile.BadZipFile) as e:
1025            1038                  zipfile.ZipFile(io.BytesIO(missing_compress_size_extra))
1026            1039              self.assertIn('compress size', str(e.exception).lower())
                1040 +
                  self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_compress_size_extra)))
1027            1041
1028            1042              # zip64 file size present, zip64 compress size present, zip64 header
1029            1043              # offset present, two fields in extra, expecting three, equals
                  missing
                  @@ -1038,6 +1052,7 @@ def test_bad_zip64_extra(self):
```

```
1038  1052          with self.assertRaises(zipfile.BadZipFile) as e:
1039  1053              zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1040  1054          self.assertIn('header offset', str(e.exception).lower())
      1055  +
                  self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
1041  1056
1042  1057          # zip64 compress size present, zip64 header offset present, one field
1043  1058          # in extra, expecting two, equals missing header offset
      @@ -1050,6 +1065,7 @@ def test_bad_zip64_extra(self):
1050  1065          with self.assertRaises(zipfile.BadZipFile) as e:
1051  1066              zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1052  1067          self.assertIn('header offset', str(e.exception).lower())
      1068  +
                  self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
1053  1069
1054  1070          # zip64 file size present, zip64 header offset present, one field in
1055  1071          # extra, expecting two, equals missing header offset
      @@ -1062,6 +1078,7 @@ def test_bad_zip64_extra(self):
1062  1078          with self.assertRaises(zipfile.BadZipFile) as e:
1063  1079              zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1064  1080          self.assertIn('header offset', str(e.exception).lower())
      1081  +
                  self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
1065  1082
1066  1083          # zip64 header offset present, no fields in extra, expecting one,
1067  1084          # equals missing header offset
      @@ -1073,6 +1090,63 @@ def test_bad_zip64_extra(self):
1073  1090          with self.assertRaises(zipfile.BadZipFile) as e:
1074  1091              zipfile.ZipFile(io.BytesIO(missing_header_offset_extra))
1075  1092          self.assertIn('header offset', str(e.exception).lower())
      1093  +
                  self.assertTrue(zipfile.is_zipfile(io.BytesIO(missing_header_offset_extra)))
      1094  +
      1095  +    def test_bad_zip64_end_of_central_dir(self):
      1096  +        zipdata = self.make_zip64_file(end_of_central_dir_size=0)
      1097  +        with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*record'):
      1098  +            zipfile.ZipFile(io.BytesIO(zipdata))
      1099  +        self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
      1100  +
      1101  +        zipdata = self.make_zip64_file(end_of_central_dir_size=100)
```

```
1102 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*record'):
1103 +             zipfile.ZipFile(io.BytesIO(zipdata))
1104 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1105 +
1106 +         zipdata = self.make_zip64_file(offset_to_end_of_central_dir=0)
1107 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*record'):
1108 +             zipfile.ZipFile(io.BytesIO(zipdata))
1109 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1110 +
1111 +         zipdata = self.make_zip64_file(offset_to_end_of_central_dir=1000)
1112 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'Corrupt.*locator'):
1113 +             zipfile.ZipFile(io.BytesIO(zipdata))
1114 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1115 +
1116 +     def test_zip64_end_of_central_dir_record_not_found(self):
1117 +         zipdata = self.make_zip64_file()
1118 +         zipdata = zipdata.replace(b"PK\x06\x06", b'\x00'*4)
1119 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'record not found'):
1120 +             zipfile.ZipFile(io.BytesIO(zipdata))
1121 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1122 +
1123 +         zipdata = self.make_zip64_file(
1124 +             extensible_data=b'\xca\xfe\x04\x00\x00\x00data')
1125 +         zipdata = zipdata.replace(b"PK\x06\x06", b'\x00'*4)
1126 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'record not found'):
1127 +             zipfile.ZipFile(io.BytesIO(zipdata))
1128 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(zipdata)))
1129 +
1130 +     def test_zip64_extensible_data(self):
1131 +         # These values are what is set in the make_zip64_file method.
1132 +         expected_file_size = 8
1133 +         expected_compress_size = 8
1134 +         expected_header_offset = 0
1135 +         expected_content = b"test1234"
1136 +
1137 +         zipdata = self.make_zip64_file(
1138 +             extensible_data=b'\xca\xfe\x04\x00\x00\x00data')
1139 +         with zipfile.ZipFile(io.BytesIO(zipdata)) as zf:
1140 +             zinfo = zf.infolist()[0]
1141 +             self.assertEqual(zinfo.file_size, expected_file_size)
```

```
        1142 +             self.assertEqual(zinfo.compress_size, expected_compress_size)
        1143 +             self.assertEqual(zinfo.header_offset, expected_header_offset)
        1144 +             self.assertEqual(zf.read(zinfo), expected_content)
        1145 +         self.assertTrue(zipfile.is_zipfile(io.BytesIO(zipdata)))
        1146 +
        1147 +         with self.assertRaisesRegex(zipfile.BadZipFile, 'record not found'):
        1148 +             zipfile.ZipFile(io.BytesIO(b'prepended' + zipdata))
        1149 +         self.assertFalse(zipfile.is_zipfile(io.BytesIO(b'prepended' +
             zipdata)))
1076    1150
1077    1151         def test_generated_valid_zip64_extra(self):
1078    1152             # These values are what is set in the make_zip64_file method.
```

Lib/zipfile/__init__.py

```
             @@ -231,41 +231,57 @@ def is_zipfile(filename):
231     231             else:
232     232                 with open(filename, "rb") as fp:
233     233                     result = _check_zipfile(fp)
234         -     except OSError:
        234 +     except (OSError, BadZipFile):
235     235             pass
236     236         return result
237     237
238     238     def _EndRecData64(fpin, offset, endrec):
239     239         """
240     240         Read the ZIP64 end-of-archive records and use that to update endrec
241     241         """
242         -     try:
243         -         fpin.seek(offset - sizeEndCentDir64Locator, 2)
244         -     except OSError:
245         -         # If the seek fails, the file is not large enough to contain a ZIP64
        242 +     offset -= sizeEndCentDir64Locator
        243 +     if offset < 0:
        244 +         # The file is not large enough to contain a ZIP64
246     245             # end-of-archive record, so just return the end record we were given.
247     246             return endrec
248         -
        247 +     fpin.seek(offset)
249     248         data = fpin.read(sizeEndCentDir64Locator)
```

```
250  249          if len(data) != sizeEndCentDir64Locator:
251       -           return endrec
     250  +           raise OSError("Unknown I/O error")
252  251          sig, diskno, reloff, disks = struct.unpack(structEndArchive64Locator,
                 data)
253  252          if sig != stringEndArchive64Locator:
254  253              return endrec
255  254
256  255          if diskno != 0 or disks > 1:
257  256              raise BadZipFile("zipfiles that span multiple disks are not
                 supported")
258  257
259       -      # Assume no 'zip64 extensible data'
260       -      fpin.seek(offset - sizeEndCentDir64Locator - sizeEndCentDir64, 2)
     258  +      offset -= sizeEndCentDir64
     259  +      if reloff > offset:
     260  +          raise BadZipFile("Corrupt zip64 end of central directory locator")
     261  +      # First, check the assumption that there is no prepended data.
     262  +      fpin.seek(reloff)
     263  +      extrasz = offset - reloff
261  264          data = fpin.read(sizeEndCentDir64)
262  265          if len(data) != sizeEndCentDir64:
263       -           return endrec
     266  +           raise OSError("Unknown I/O error")
     267  +      if not data.startswith(stringEndArchive64) and reloff != offset:
     268  +          # Since we already have seen the Zip64 EOCD Locator, it's
     269  +          # possible we got here because there is prepended data.
     270  +          # Assume no 'zip64 extensible data'
     271  +          fpin.seek(offset)
     272  +          extrasz = 0
     273  +          data = fpin.read(sizeEndCentDir64)
     274  +          if len(data) != sizeEndCentDir64:
     275  +              raise OSError("Unknown I/O error")
     276  +      if not data.startswith(stringEndArchive64):
     277  +          raise BadZipFile("Zip64 end of central directory record not found")
     278  +
264  279          sig, sz, create_version, read_version, disk_num, disk_dir, \
265  280              dircount, dircount2, dirsize, diroffset = \
266  281              struct.unpack(structEndArchive64, data)
267       -      if sig != stringEndArchive64:
```

```
268      -          return endrec
         282 +      if (diroffset + dirsize != reloff or
         283 +          sz + 12 != sizeEndCentDir64 + extrasz):
         284 +          raise BadZipFile("Corrupt zip64 end of central directory record")
269      285
270      286        # Update the original endrec using data from the ZIP64 record
271      287        endrec[_ECD_SIGNATURE] = sig
         @@ -275,6 +291,7 @@ def _EndRecData64(fpin, offset, endrec):
275      291        endrec[_ECD_ENTRIES_TOTAL] = dircount2
276      292        endrec[_ECD_SIZE] = dirsize
277      293        endrec[_ECD_OFFSET] = diroffset
         294 +      endrec[_ECD_LOCATION] = offset - extrasz
278      295        return endrec
279      296
280      297
         @@ -308,7 +325,7 @@ def _EndRecData(fpin):
308      325            endrec.append(filesize - sizeEndCentDir)
309      326
310      327            # Try to read the "Zip64 end of central directory" structure
311      -          return _EndRecData64(fpin, -sizeEndCentDir, endrec)
         328 +          return _EndRecData64(fpin, filesize - sizeEndCentDir, endrec)
312      329
313      330        # Either this is not a ZIP file, or it is a ZIP file with an archive
314      331        # comment.  Search the end of the file for the "end of central directory"
         @@ -332,8 +349,7 @@ def _EndRecData(fpin):
332      349            endrec.append(maxCommentStart + start)
333      350
334      351            # Try to read the "Zip64 end of central directory" structure
335      -          return _EndRecData64(fpin, maxCommentStart + start - filesize,
336      -                               endrec)
         352 +          return _EndRecData64(fpin, maxCommentStart + start, endrec)
337      353
338      354        # Unable to find a valid end of central directory structure
339      355        return None
         @@ -1427,9 +1443,6 @@ def _RealGetContents(self):
1427     1443
1428     1444            # "concat" is zero, unless zip was concatenated to another file
1429     1445            concat = endrec[_ECD_LOCATION] - size_cd - offset_cd
```

```
1430        -         if endrec[_ECD_SIGNATURE] == stringEndArchive64:
1431        -             # If Zip64 extension structures are present, account for them
1432        -             concat -= (sizeEndCentDir64 + sizeEndCentDir64Locator)
1433   1446
1434   1447           if self.debug > 2:
1435   1448               inferred = concat + offset_cd
```

```
@@ -2047,7 +2060,7 @@ def _write_end_record(self):
2047   2060                                       " would require ZIP64 extensions")
2048   2061               zip64endrec = struct.pack(
2049   2062                   structEndArchive64, stringEndArchive64,
2050        -                  44, 45, 45, 0, 0, centDirCount, centDirCount,
        2063 +                  sizeEndCentDir64 - 12, 45, 45, 0, 0, centDirCount,
 centDirCount,
2051   2064                   centDirSize, centDirOffset)
2052   2065               self.fp.write(zip64endrec)
2053   2066
```

...5-10-07-19-31-34.gh-issue-139700.vNHU1O.rst

```
@@ -0,0 +1,3 @@
1  + Check consistency of the zip64 end of central directory record. Support
2  + records with "zip64 extensible data" if there are no bytes prepended to the
3  + ZIP file.
```

## Comments 0